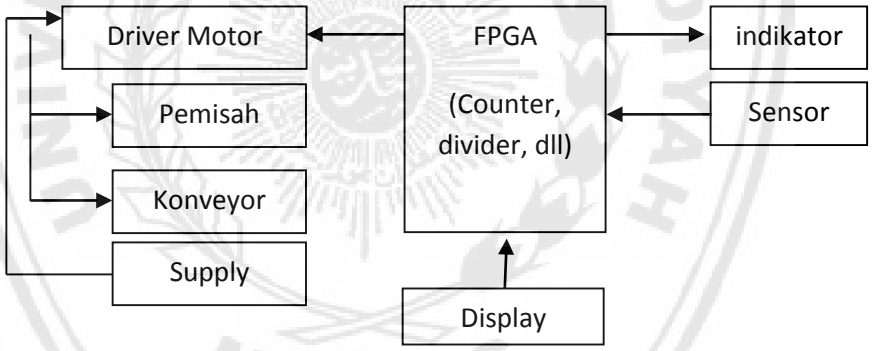
# BAB III
# PERANCANGAN DAN PEMBUATAN ALAT

Konveyor merupakan suatu perangkat dalam dunia industrI yang selalu dipakai sebagai saran transportasi dari satu system ke system lainnya dengan control PLC sebagai pusat otomasi keseluruhan system. PLC merupakan perangkat yang berisi relay-relay yang terintegrasi dengan instruksi-instruksi logic, hal ini memiliki system yang hamper sama dengan FPGA yang merupakan system elektronika digital.

## 3.1 Konsep Sistem Kontrol FPGA

Pada perancangan system ini menggunakan berbagai modul yang saling berhubungan antara satu dengan yang lainnya, dengan FPGA sebagai pusat control konveyor pemilih barang berdasarkan ketinggian kemasan. Diagram blok system control FPGA ditunjukkan pada gambar dibawah ini :



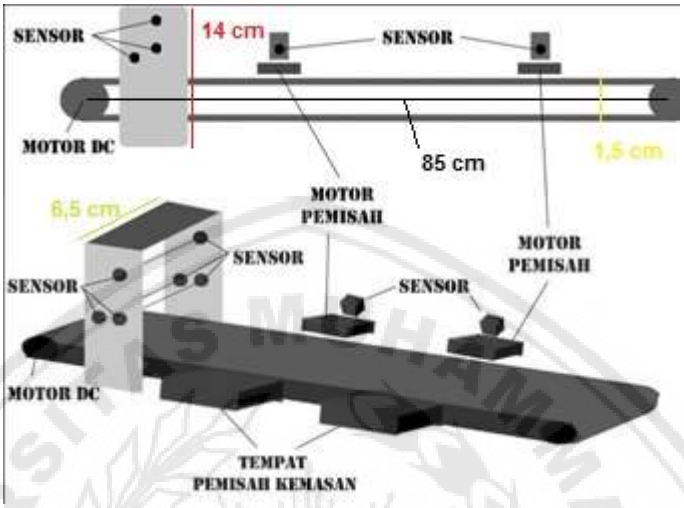

Gambar 3.1 Diagram Blok Kontrol FPGA

Perangkat yang digunakan IC FPGA yang merupakan salah satu *embedded system* yang dapat diaplikasikan terhadapa elektronika digital dimana meliputi, frequency divider dimana berfungsi sebagai pembagi frequency, hal ini diperlukan karena crystal pendetak yang digunakan sebesar 50 MHz. selain itu juga menggunakan konsep dasar *counter* baik sebagai penghitung maupun sebagai *driver delay* pada pemakaian bahasa pemrogaman FPGA.

## 3.2 Konsep Mekanik Konveyor

Mekanik merupakan salah satu hal yang sangat penting dalam merancang suatu system. Dalam merancang suatu alat yang berupa *hardware*, mekanik

merupakan salah satu hal yang penting karena menunjang kelancaran suatu system. Dalam perancangan mekanik pada system ini menggunakan conveyor yang memanfaatkan motor sebagai pemutar belt conveyor. Selain motor pemutar belt conveyor juga terdapat motor pendorong yang terintegrasi dengan sensor sebagai penentu posisi barang. Desain mekanik dapat dilihat pada gambar dibawah ini :




Gambar 3.2 Desain Mekanik Conveyor

## 3.3 Konsep Rancangan Perangkat Keras

Perancangan perangkat keras pasa system conveyor pemilih barang berdasarkan ketinggian kemasan didukung oleh beberapa beberapa perangkat lain yang saling terhubung yaitu sebagai berikut :

1) Perangkat control yang digunakan adalah FPGA SPARTAN dari vendor Xilinx.
2) Perangkat pendeteksi ketinggian barang menggunakan sensor IR (Inframerah) dan photodiode.
3) IC driver sebagai pengontrol motor dc pada konveyor dan motor eksekusi barang.
4) Relay sebagai media bantu dalam proses pensaklaran terhadap interface lainnya.

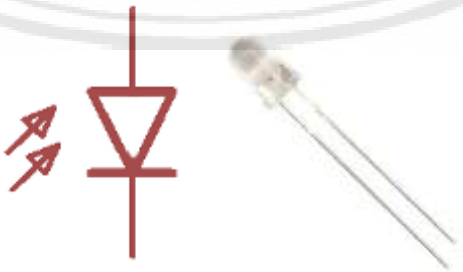
## 3.4 Perancangan Sensor

Perangkat sensor memanfaatkan led IR dan photodiode sebagai sensor deteKtor. Sensor inframerah itu sendiri adalah radiasi elektromagnetik dari panjang gelombang lebih panjang dari cahaya tampak, tetapi lebih pendek dari radiasi

gelombang radio. Namanya berarti "bawah merah" (dari bahasa Latin infra, "bawah"), merah merupakan warna dari cahaya tampak dengan gelombang terpanjang. Radiasi inframerah memiliki jangkauan tiga "order" dan memiliki panjang gelombang antara 700 nm dan 1 mm. Inframerah ditemukan secara tidak sengaja oleh Sir William Herschell, astronom kerajaan Inggris ketika ia sedang mengadakan penelitian mencari bahan penyaring optik yang akan digunakan untuk mengurangi kecerahan gambar matahari dalam tata surya teleskop. Sedangkan pembangkit sinar infrared ini adalah sebuah dioda LED. Secara kasat mata cahaya infrared dapat dilihat dengan memanfaatkan kamera *handphone*.

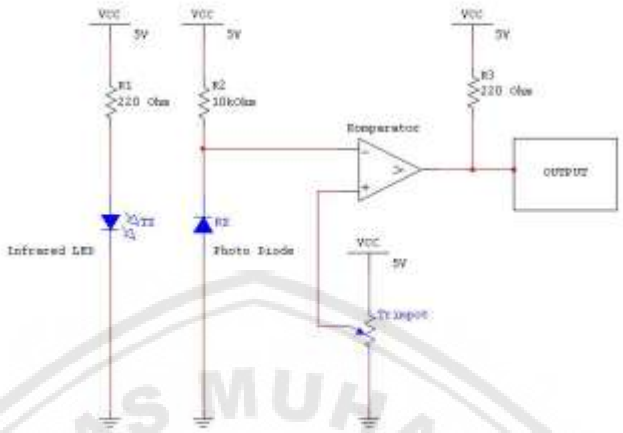
Gambar 3.3 Sensor Led IR(Inframerah)

Photodioda adalah dioda yang bekerja berdasarkan intensitas cahaya, dimana jika photodioda terkena cahaya maka photodioda bekerja seperti dioda pada umumnya, tetapi jika tidak mendapat cahaya maka photodioda akan berperan seperti resistor dengan nilai tahanan yang besar sehingga arus listrik tidak dapat mengalir.



Gambar 3.4 Komponen Photodioda

dengan memanfaatkan kedua interface tersebut dapat dijadikan suatu perangkat yang mampu mendeteksi ada tidaknya suatu barang yang melewati area tersebut. Seperti rancangan dibawah ini :




Gambar 3.5 Rancangan Sensor Detektor Barang

Pada rancangan diatas dapat dilihat bahwa sensor IR digunakan sebagai pengirim sinyal dan photodiode sebagai penerima sinyal. Pada rancangan diatas dapat dijelaskan saat intensitas Infrared yang diterima Photodiode besar maka tahanan Photodiode menjadi kecil, sedangkan jika intensitas Infrared yang diterima Photodiode kecil maka tahanan yang dimiliki photodiode besar. Jika tahanan photodiode kecil maka tegangan V- akan kecil. Misal tahanan photodiode mengecil menjadi 10kOhm. Maka dengan rumus pembagi tegangan sebagai berikut :

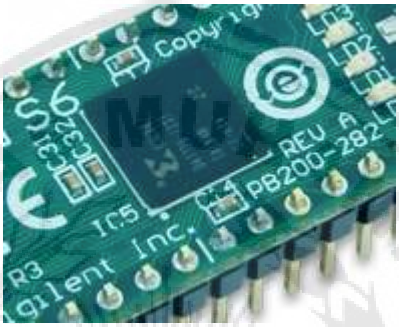
$$V- = \frac{R_{rx}}{R_{rx} + R_2} \; x \; V_{cc}$$

$$= \frac{10K}{10K + 10K} \; x \; 5 = 2{,}5 \; V$$

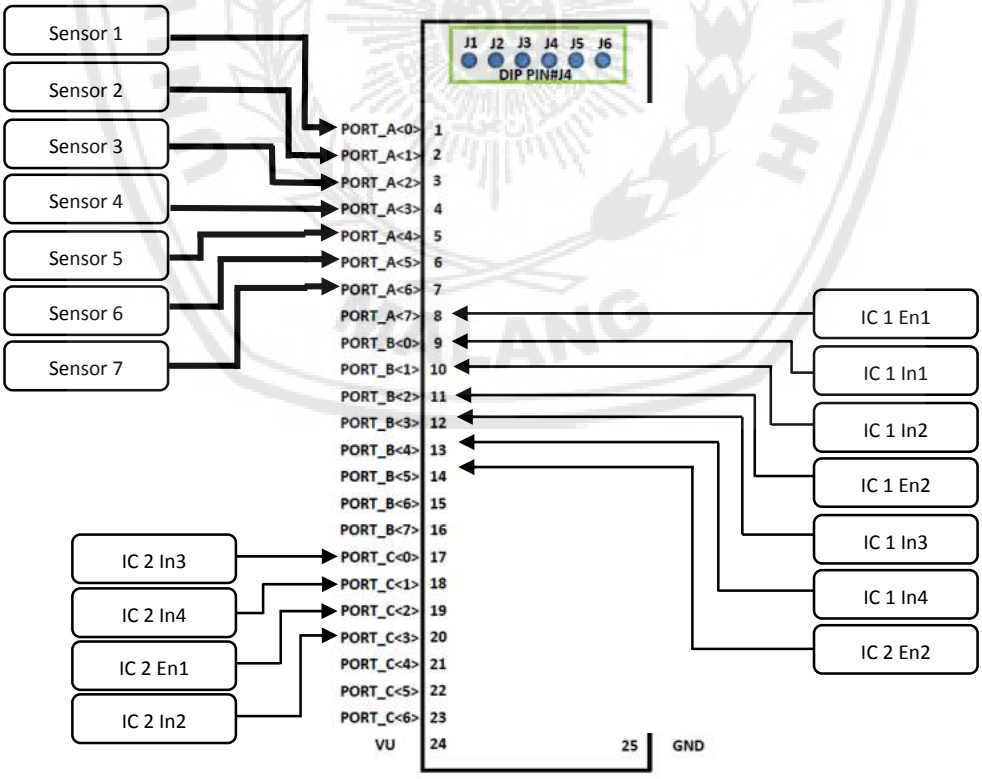
Keluaran tersebut masuk ke komperator dimana meupakan pembanding untuk menentukan setpoint yang dikeluarkan dengan mengubah nilai pada trimpot. Jika nilai V+ lebih dari V- maka keluaran komperator = 5 V, jika kurang dari V- maka keluaran komperator = 0 V.

### 3.5 Perancangan dan Pembuatan Perangkat Kontrol

Perancangan perangkat pada control ini mengunakan FPGA Xilinx Spartan dimana perangkat yang banyak digunakan dalam suatu system yang termasuk *SoC (System on Chip)*. Perangkat ini menggunakan landasan dasar elektronika digital yang saling terintegrasi dan memiliki eksekusi perintah yang terbilang cepat. Spartan  dan pin I/O dapat dilihat [ada gambar dibawah ini :




Gambar 3.6 FPGA SPARTAN 6




Gambar 3.7 Pin I/O SPARTAN 6

Pin I/O yang terdapat pada perangkat tersebut memiliki 48 pin yang dapat dimanfaatkan sebagai penghubung interface lainnya. Selain itu juga terdapat detak sebesar 32MHz yang dimana speed ini mampu diolah kembali dengan memanfaatkan fasilitas *Digital Clock Manager (DCM)* sehingga mampu dalam lingkup 50 MHz. Pin I/O ini mampu digunakan oleh keseluruhan system dengan menggabungkan kebutuhan elektronika, system bekerja pada tegangan 3,3 Volt sesuai dengan kebutuhan system yang berbasis pada *logic gate.*

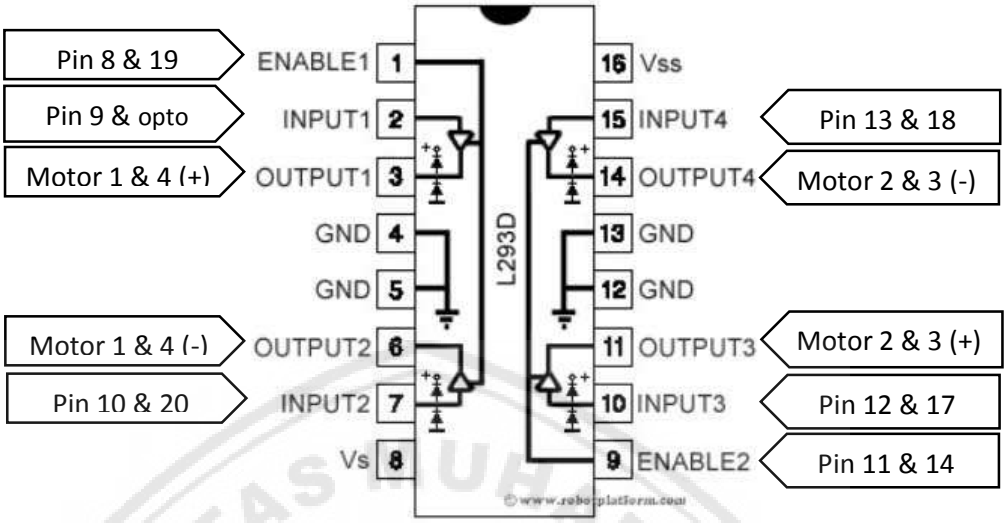
### 3.6 Konsep rancangan *driver* motor

Pada perancangan driver motor menggunakan ic driver L293D sebagai pengganti driver motor menggunakan transistor. Driver ini diperlukan supaya motor mampu bergerak maju maupun bergerak mundur. Motor DC sendiri merupakan jenis motor yang menggunakan tegangan searah sebagai sumber tenaganya. Dengan memberikan beda tegangan pada kedua terminal tersebut, motor akan berputar pada satu arah, dan bila polaritas dari tegangan tersebut dibalik maka arah putaran motor akan terbalik pula. Polaritas dari tegangan yang diberikan pada dua terminal menentukan arah putaran motor sedangkan besar dari beda tegangan pada kedua terminal menentukan kecepatan motor.

Gambar 3.8 Motor DC

Sedangkan IC L293D adalah IC yang didesain khusus sebagai driver motor DC dan dapat dikendalikan dengan rangkaian TTL maupun mikrokontroler. Motor DC yang dikontrol dengan driver IC L293D dapat dihubungkan ke ground maupun ke sumber tegangan positif karena di dalam driver L293D sistem driver yang digunakan adalah totem pool. Dalam 1 unit chip IC L293D terdiri dari 4 buah driver motor DC yang berdiri sendiri sendiri dengan kemampuan mengalirkan arus 1 Ampere tiap drivernya. Sehingga dapat digunakan untuk membuat driver H-bridge

untuk 2 buah motor DC. Konstruksi pin driver motor DC IC l293D adalah sebagai berikut.




Gambar 3.9 Konstruksi IC L293D

## 3.7 Perancangan Sistem Register Data Sensor

Perancangan sistem register data ini memiliki peranan penting dimana data yang dihasilkan dari sensor akan dikirim ke register untuk diolah oleh sistem-sistem yang lain, hal ini bertujuan agar data dari sensor tidak mengalami perubahan akibat *shock operation.*

- Bahasa VHDL Data Sensor

```
entity sensor is
   Port ( inpdat : in  STD_LOGIC_VECTOR (3 downto 1);
          clk1 : in  STD_LOGIC; --
          outd : out  STD_LOGIC_VECTOR (3 downto 1));
end sensor;

architecture Behavioral of sensor is
signal temp : std_logic_vector (3 downto 1):= "000"; begin
process (clk1)
  begin
          if (clk1'event and clk1='1') then
          case inpdat is
          when "001" => temp <= "001";
          when "011" => temp <= "010";
          when "111" => temp <= "100";
          when others => temp <= "000";
          end case;
  end if;
```

```
end process;
  outd <= temp;
```

- Bahasa VHDL Register

```
entity Regtsr is
port (
load : in std_logic;
inpt : in std_logic_vector (3 downto 1);
clk  : in std_logic;
clr  : in std_logic;
oupt : out std_logic_vector (3 downto 1));
end Regtsr;

architecture Behavioral of Regtsr is

begin
process(clk, clr)
begin
        if clr = '1' then
                oupt <= "000";
        elsif clk'event and clk='1' then
                if load ='1' then
                        oupt <= inpt;
                end if;
        end if;
end process;
end Behavioral;
```

- Bahasa VHDL *Top Level* Sistem Sensor

```
entity sens_reg is
   Port ( clk : in  STD_LOGIC;
                clr : in  STD_LOGIC;
          sensor_in : in  STD_LOGIC_VECTOR (3 downto 1);
          load_sensor : in  STD_LOGIC;
          out_sensor : out  STD_LOGIC_VECTOR (3 downto 1));
end sens_reg;

architecture Behavioral of sens_reg is
component sensor is
   Port ( inpdat : in  STD_LOGIC_VECTOR (3 downto 1);
          clk1 : in  STD_LOGIC; --
          outd : out  STD_LOGIC_VECTOR (3 downto 1));
end component;

component Regtsr is
port (
```

```
load : in std_logic;
inpt : in std_logic_vector (3 downto 1);
clk  : in std_logic;
clr  : in std_logic;
oupt : out std_logic_vector (3 downto 1));
end component;

signal temp : std_logic_vector (3 downto 1);
begin
x1 : sensor port map (
                clk1=>clk,
                inpdat=>sensor_in,
                outd=>temp);

x2 : Regtsr port map (
                clk=>clk,
                clr=>clr,
                load=>load_sensor,
                inpt=>temp,
                oupt=>out_sensor);

end Behavioral;
```

### 3.8 Perancangan Sistem Motor Pendorong

Pada perancangan ini sistem menggunakan register sebagai tempat penyimpanan perubahan data yang terjadi selain itu tersebut juga dapat digunakan untuk mengontrol sistem lainnya.

```
PORT( clk : IN  STD_LOGIC;
        load    : IN  STD_LOGIC;
        ind1  : IN std_logic;
        rst   : IN  STD_LOGIC;
        out_rst : out  STD_LOGIC;
        motor1 : OUT STD_LOGIC_VECTOR(3 DOWNTO 1)
        );

end motor_one;

architecture Behavioral of motor_one is

CONSTANT div : INTEGER :=3999999;

SIGNAL clk_div : INTEGER RANGE 0 TO div := 0;
SIGNAL temp : INTEGER RANGE 0 TO 5 := 0;
signal temp1 : std_logic_vector (4 downto 1):="0000";
```

```
begin

PROCESS(clk)
BEGIN
IF RISING_EDGE(clk) THEN
  IF rst = '1' THEN
          clk_div <= 0;
          temp <= 0;
  ELSE
          IF ind1 = '1' THEN
                  IF load = '1' then
                  clk_div <= clk_div + 1;
                          IF clk_div = div THEN
                                  clk_div <= 0;
                                  temp <= temp +1;
                                          IF temp = 5 THEN
                                                  temp <= 0;
                                          END IF;
                          END IF;
                  END IF;
          END IF;
  END IF;
END IF;
END PROCESS;

  temp1 <= "0000" WHEN temp = 0 ELSE
                  "0011" WHEN temp = 1 ELSE
                  "0000" WHEN temp = 2 ELSE
                  "0101" WHEN temp = 3 ELSE
                  "0000" WHEN temp = 4 ELSE
                  "1000" WHEN temp = 5 ELSE
                  "0000";
motor1(1)<=temp1(1);
motor1(2)<=temp1(2);
motor1(3)<=temp1(3);
out_rst<=temp1(4);
end Behavioral;
```

### 3.9 Perancangan Sistem Motor Konveyor

Pada perancangan ini memerlukan suatu sistem dimana motor konveyor memiliki kondisi berhenti sejenak untuk menyesuaikan posisi barang agar dapat dieksekusi oleh sistem yang terintegrasi.

```
entity kov_motor is
   Port ( clk : in  STD_LOGIC;
        motor_kov : out  STD_LOGIC_VECTOR (6 downto 5):="01";
```

```
        data_sens : in  STD_LOGIC_VECTOR (10 downto 5);
        clr : in  STD_LOGIC);
end kov_motor;

architecture Behavioral of kov_motor is

begin
process(clk, clr)
begin
  if clr = '1' then
  motor_kov <= "01";
          elsif (clk'event and clk='1') then
                  case data_sens is
                          when "001001" => motor_kov <= "00";
                          when "010011" => motor_kov <= "00";
                          when "100111" => motor_kov <= "00";
                          when others => motor_kov <= "10";
                  end case;
  end if;
end process;
end Behavioral;
```

### 3.10 Perancangan *Top Level System*

Pada perancangan ini diperlukan untuk menghubungkan keseluruhan sistem yang telahdibuat menggunakan *mode architecture structural*, penggunaan mode ini karena dapat mengetahui letak error jika sistem mendapati error dengan melihat bagian RTL sistem.

```
entity konveyor_sistem is
   Port ( clkk : in  STD_LOGIC;
        clrk : in  STD_LOGIC;
        load_konv1 : in  STD_LOGIC;
                load_konv2 : in  STD_LOGIC;
                load_konv3 : in  STD_LOGIC;
                load_sensor : in std_logic;
                sensor_barang : in  STD_LOGIC_VECTOR (3 downto 1);
        out_motork1 : out  STD_LOGIC_VECTOR (3 downto 1);
        out_motork2 : out STD_LOGIC_VECTOR (3 downto 1);
        out_motork3 : out  STD_LOGIC_VECTOR (3 downto 1);
        motor_konveyor : out STD_LOGIC_VECTOR (6 downto 5));
end konveyor_sistem;

architecture Behavioral of konveyor_sistem is
component sens_m1 is
port (
  loadm1 : in std_logic;
```

```
  clk  : in std_logic;
  clr  : in std_logic;
  oupm1 : out std_logic);
end component;

component sens_m2 is
port (
  loadm2 : in std_logic;
  clk  : in std_logic;
  clr  : in std_logic;
  oupm2 : out std_logic);
end component;

component sens_m3 is
port (
  loadm3 : in std_logic;
  clk  : in std_logic;
  clr  : in std_logic;
  oupm3 : out std_logic);
end component;

component bridge is
port ( clk : in std_logic;
           in1 : in std_logic:='0';
           in2 : in std_logic:='0';
           in3 : in std_logic:='0';
           inpen1 :in std_logic:='0';
           inpen2 :in std_logic:='0';
           inpen3 :in std_logic:='0';
           out_dat : out std_logic_vector (6 downto 1):="000000");
end component;

component kov_motor is
   Port ( clk : in  STD_LOGIC;
          motor_kov : out  STD_LOGIC_VECTOR (6 downto 5);
          data_sens : in  STD_LOGIC_VECTOR (10 downto 5):="000000";
          clr : in  STD_LOGIC);
end component;

component data_kov_clr is
   Port ( data_in : in  STD_LOGIC;
          data_out: out  STD_LOGIC;
          clk : in  STD_LOGIC;
          clr : in  STD_LOGIC);
end component;

component pendorong is
  Port ( clkp : in  STD_LOGIC;
```

```
        clrp : in  STD_LOGIC;
                    rst : in std_logic;
                    load_sel : in std_logic;
                    sensor : in  STD_LOGIC_VECTOR (3 downto 1);
                    loadm1 : in std_logic;
                    loadm2 : in std_logic;
                    loadm3 : in std_logic;
                    rst_out : out std_logic;
                    data_bridge1 : out std_logic:='0';
                    data_bridge2: out std_logic:='0';
                    data_bridge3 : out std_logic:='0';
        out_motor1 : out  STD_LOGIC_VECTOR (3 downto 1);
        out_motor2 : out STD_LOGIC_VECTOR (3 downto 1);
        out_motor3 : out  STD_LOGIC_VECTOR (3 downto 1));
end component;

component switch is
   Port ( sw : in  STD_LOGIC;
        ot : out  STD_LOGIC);
end component;


signal temp2 : std_logic;
signal a,b,c,d : std_logic;
signal clr_sig : std_logic;
signal data_mot1 : std_logic_vector (3 downto 1):= "000";
signal pen1, pen2, pen3 : std_logic := '0';
signal temp : std_logic_vector (6 downto 1):= "000000";


begin

x1 : sens_m1 port map (
                    clk=>clkk,
                    clr=>clr_sig,
                    loadm1=>load_konv1,
                    oupm1=>a);

x2 : sens_m2 port map (
                    clk=>clkk,
                    clr=>clr_sig,
                    loadm2=>load_konv2,
                    oupm2=>b);

x3 : sens_m3 port map (
                    clk=>clkk,
                    clr=>clr_sig,
                    loadm3=>load_konv3,
```

```
                    oupm3=>c);

x4 : bridge port map (
            clk=>clkk,
            in1=>a,
            in2=>b,
            in3=>c,
            inpen1=>pen1,
            inpen2=>pen2,
            inpen3=>pen3,
            out_dat=>temp);

x5 : kov_motor port map (
                    clk=>clkk,
                    clr=>clrk,
        motor_kov=>motor_konveyor,
        data_sens=>temp);

x6 : data_kov_clr port map (
                    clk=>clkk,
                    clr=>clrk,
        data_in=>temp2,
        data_out=>clr_sig);

x7 : pendorong port map (
                    clkp=>clkk,
         clrp=>d,
                    load_sel=>load_sensor,
                    sensor=>sensor_barang,
                    rst=>d,
                    loadm1=>a,
                    loadm2=>b,
                    loadm3=>c,
                    rst_out=>temp2,
                    data_bridge1=>pen1,
                    data_bridge2=>pen2,
                    data_bridge3=>pen3,
          out_motor1=>out_motork1,
          out_motor2=>out_motork2,
          out_motor3=>out_motork3);

x8 : switch port map(
                    sw=>temp2,
                    ot=>d);

end Behavioral;
```